J Netw Syst Manage (2016) 24:916–943
DOI 10.1007/s10922-015-9357-5

Cloud Service Recommendation Based on a Correlated QoS Ranking Prediction

K. Jayapriya[1] · N. Ani Brown Mary[2] · R. S. Rajesh[3]

Received: 13 February 2015 / Revised: 12 October 2015 / Accepted: 14 October 2015 /
Published online: 14 November 2015

Abstract Quality-of-Service (QoS) is an important concept for service selection and user satisfaction in cloud computing. So far, service recommendation in the cloud is done by means of QoS, ranking and rating techniques. The ranking methods perform much better, when compared with the rating methods. In view of the fact that the ranking methods directly predict QoS rankings as accurately as possible, in most of the ranking methods, an individual QoS value alone is employed to predict the cloud rank. In this paper, we propose a correlated QoS ranking algorithm along with a data smoothing technique and combined with QoS to predict a personalized ranking for service selection by an active user. Experiments are conducted employing a WSDream-QoS dataset, including 300 distributed users and 500 real world web services all over the world. Six different techniques of correlated QoS ranking schemes have been proposed and evaluated. The experimental results showed that this approach improves the accuracy of ranking prediction when compared to a ranking prediction framework using a single QoS parameter.

Keywords Correlation coefficient · Correlated QoS ranking prediction · Multiple QoS · Data smoothing · Response time · Throughput

✉ K. Jayapriya
kjp.jayapriya@yahoo.com

N. Ani Brown Mary
anibrownvimal@gmail.com

R. S. Rajesh
rsrajesh_cse@msuniv.ac.in

[1] Vin Solutions, 1st Floor, No 40, North Street, Rajarajeshwari Nagar, Tirunelveli 627007, India

[2] Department of Computer Science, Regional Centre of Anna University, Tirunelveli 627007, India

[3] Department of Computer Science, M.S. University, Tirunelveli 627012, India

1 Introduction

The outbreak of cloud computing has provoked a vital turn in the prospects of research and business organizations in IT infrastructures. It is the next step in the development of information technology services and products. Despite spending a huge price on hardware and plenty of money on maintenance costs, IT organizations can be preferred with little cost. The evidence of increases in the web services of companies such as Amazon [1, 19, 35], Google [18] and Salesforce [44, 47] dramatically shows how cloud computing is desirable in recent times. Amazon.com is one of the most important and heavily trafficked web sites in the world. It provides a vast selection of products using an infrastructure based on web services. Google is the prototypical cloud computing services company, and it supports some of the largest web sites and services in the world.Salesforce.com is a web application suite that is 'Software as a Service' (SaaS) and Force.com is Salesforce.com's 'Platform as a Service' (PaaS) platform for building one's own services.

Due to the growth of public cloud contributions, cloud consumers have become progressively more difficult to decide which provider can fulfill their Quality of Service (QoS) requirements. "QoS is the service providers' capability to achieve the service users' requirements, such as response time, throughput, availability, security and so forth". Each cloud provider offers similar services at various prices and performance levels with different sets of features. Due to the multiplicity of cloud service offerings, an important issue that the consumers are concerned with is how to discover who are the "right" cloud providers that can satisfy their requirements. Therefore, it is not sufficient to just discover multiple cloud services but it is also significant to evaluate which is the most suitable cloud service.

In conventional component-based systems, software components are invoked in the vicinity. The client-side web service manipulations need real-time web service implementations and endeavours the following shortcomings: First, the reality of web service implementations compels costs for the user and intake resources from the service providers. Few of the initiations may be charged. Secondly, too many service applicants must be manipulated and few may not even be recognized in the lists. All users of web services may not be well-versed or highly trained in web service manipulation. The trivial time-to-market constraints restrict a manipulation of the aimed web services.

The general idea is that the QoS values of all candidate services to target users are known. On the other hand, it might not be true in realism. Owing to some factors, e.g., location and network environment, the QoS of the same service to different users may be different. For example, the response time for user (IP:12.108.128.196, RUSSIA) to invoke Web service (WSDL:http://biomoby.org/services/wsdl/mmb.pcb.ub.es/parseFeatureAASequenceFromFSOLVText, located in USA) is 5916 ms; whereas that for user (IP:183.1.74.162, Kenya) to invoke the same service is 690 ms. A user can barely invoke all services, meaning that the QoS values of the services that the user has not invoked are unknown. Hence, an adapted cloud service QoS ranking is required for different cloud applications.

Nonetheless, most of these existing methods [22, 57, 59] focus on the method of finding similarity between users and their services and then to find the missing value prediction of the users. The most undemanding approach of an adapted cloud service QoS ranking is to estimate all the applicant services at the user's side and rank the services based on the observed QoS values. However, this approach is impossible in reality, since invocations of cloud services may be charged. Additionally, when the quantity of applicant services is large, it is not easy for the cloud application trend to evaluate all the cloud services professionally.

To overcome this critical challenge, Zheng [58] proposed the first step on a personalized ranking prediction framework for a current user. This approach predicts the QoS ranking of a set of cloud services, even though some services are not invocated by the current user. The author proposed two ranking prediction algorithms for computing the service ranking based on the cloud application designer's preferences. Those two ranking algorithms perform well compared to the traditional greedy [31] and rating based [17] approaches. However, in this approach, the author predicts the rank on the basis of a single QoS value. So in this case, sometimes it may provide a different rank position for the same service based on different QoS parameters. Hence, we need to provide a single personalized ranking by using correlation properties of combined QoS values. This concept motivates us to produce a correlated QoS ranking for cloud services to improve the ranking accuracy.

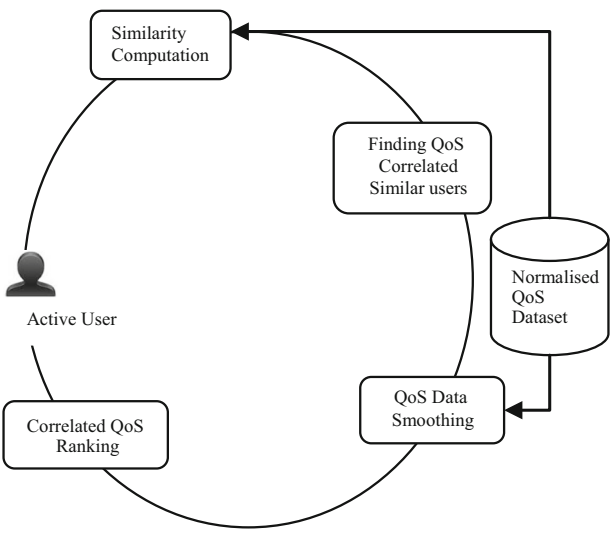


Fig. 1 System architecture of CorQoSCloudRank

2 System Overview

QoS is an essential idea in ranking process for users in cloud computing. This paper mainly focuses on a correlated ranking of user side properties, which are likely to have different values for different users of the same cloud service. More accurate correlated QoS ranking results can be accomplished by providing QoS values on additional cloud services, because the feature of an active user can be extracted from the given information. This paper focuses on examining the response time and throughput of different web services and service users. Response time is defined as the time taken between a service user sending a request and receiving the corresponding response. Throughput is defined as the average rate of successful service delivery.

Figure 1 depicts the system architecture of our CorQoSCloudRank framework, which provides correlated QoS ranking for cloud services. Here, the service users who require QoS services are named as active or current users. When an active user's obligation arrives, the process of finding correlated similar users is first engaged. In the procedure of finding correlated similar users, we will compute similarity computation with the help of the Pearson Correlation Coefficient (PCC), Spearman Rank Correlation Coefficient (SRCC) and Kendall Rank Correlation Coefficient (KRCC) using Normalized QoS values. Here, the datasets consist of Normalized QoS values. After finding the QoS correlation of similar users, we employ a data smoothing technique in normalized QoS datasets. This data smoothing is an efficient technique that is used to improve the accuracy of QoS ranking. For data smoothing, we employ a Fuzzy-C-Means (FCM) algorithm. A FCM algorithm is generally characterized as either a grouping of similar data values around a center or a prototype data instance nearest to the centered. Then, the preference function is estimated. During this process, the unknown QoS values of each and every user will be predicted with the help of the QoS correlated similar users. On the basis of estimated preference function values with data smoothing, correlated QoS ranking is employed.

The framework introduced in this work improves the accuracy of the QoS ranking with the help of a correlated QoS ranking technique. Correlated QoS ranking is implemented using an algorithm called CorQoSCloudRank. With the help of this algorithm, QoS ranking for all services has been ranked in an efficient way.

3 Correlated Qos Ranking Methodology

This section presents our Correlated QoS personalized ranking prediction framework for cloud services. Section 3.1 calculates the similarity of the active user with the training users based on correlated QoS values of commonly invoked cloud services. Section 3.2 describes the data smoothing concept that improves the ranking accuracy of our approach. Section 3.3 presents the description of the

proposed Correlated QoS ranking prediction algorithm, named CorQoSCloudRank, respectively. Section 3.4 analyzes the ranking accuracy of the proposed approach.

3.1 Finding QoS Correlated Similar Users

This section commences the similarity estimation method of various service users. Let R be the user-service realistic QoS matrix of the size M × N where M symbolizes the number of users and N symbolizes number of services. At this instant, I symbolizes a set of services that is I = $\{i_1, i_2, i_3,\ldots,i_N\}$ and U symbolizes a set of users or (user set) that is U = $\{u_1, u_2, u_3,\ldots,u_M\}$ Each entry in this matrix $R_{a,i}$ represents a vector of QoS values that is observed by the service user a on the service item i. If user a did not invoke the service item i in his previous transactions, then $R_{a,i}$ = null.

Similarity estimation is very useful to identify users utilizing the same resources. It is applied to compute the similarity between the users who use the same type of cloud resources and also to compute similarity among users who apply at least some of the resources. The relationship among user similarity is denoted by an M × M matrix, called the user–user similarity matrix. Similarity values generally ranges from 0 to 1, where 1 signifies an absolute value and 0 signifies an null value. Table 1 clearly depicts the similarity values and their variables. In this approach, three types of similarity measures are well-known. They are the PCC, SRCC and KRCC.

First, let us deal with the PCC [21, 22] that measures the similarity between two users based on their normalized services as;

$$SIM(a,b) = \frac{\sum_{\mathrm{i}\in \mathrm{I}a\cap \mathrm{I}_b}(\mathrm{r}_{a,\mathrm{i}} - \bar{\mathrm{r}}_a)(\mathrm{r}_{b,\mathrm{i}} - \bar{\mathrm{r}}_b)}{\left[\sum_{\mathrm{i}\in \mathrm{I}a\cap \mathrm{I}_b}(\mathrm{r}_{a,\mathrm{i}} - \bar{\mathrm{r}}_a)^2 \sum_{\mathrm{i}\in \mathrm{I}a\cap \mathrm{I}_b}(\mathrm{r}_{b,\mathrm{i}} - \bar{\mathrm{r}}_b)^2\right]^{1/2}}, \quad (1)$$

where a, b symbolize users, i, j symbolize services, $I_a \cap I_b$ is the subset of cloud services commonly invoked by users a and b, $r_{a,i}$ is the resources of service 'i' observed by user a, $r_{a,i}$ is the resources of service 'i' observed by user b, $\bar{r}_a, \bar{r}_b$ is the average of resources worn by users a and b. From this description, the similarity of two service users, $SIM(a, b)$, is in the interval of [−1,1], where a larger Pearson value indicates that service users a and b are more similar [59]. User-based

Table 1 Similarity values and variables

Similarity value	Similarity variable
1	Accurate
0.9–1	Very high
0.7–0.9	High
0.4–0.7	Medium
0.2–0.4	Low
0–0.2	Very low
0	None

collaborative filtering by means of PCC was employed in quite a lot of recommended [22, 43].

Second, the SRCC measures the strength of association between two ranked variables. It measures the similarity between two users a and b such as;

$$SIM(a,b) = \frac{\sum_{\mathrm{i}} (a_{\mathrm{i}} - \bar{a})(b_{\mathrm{i}} - \bar{b})}{\sqrt{\sum_{\mathrm{i}} (a_{\mathrm{i}} - \bar{a})^2 \sum_{\mathrm{i}} (b_{\mathrm{i}} - \bar{b})^2}}, \tag{2}$$

where 'i' symbolizes service, a_i symbolizes the ith service taken by ath user and b_i represents the ith service taken by bth user, $\bar{a}, \bar{b}$ represents the average users who avail the service. When two service users have a null service intersection, the value of SIM(a,b) cannot be computed ($SIM(a,b) =$ null). If there are no repeated values, a perfect Spearman correlation of $+1$ or -1 occurs when each of the variables is a perfect monotone function of the other.

Third, the KRCC measures the similarity between two service rankings,

$$SIM(a,b) = 1 - \frac{4 \times \sum_{i,j \in I_a \cap I_b} \tilde{I}((r_{a,i} - r_{a,j})(r_{b,i} - r_{b,j}))}{|I_a \cap I_b| \times |I_a \cap I_b| - 1} \tag{3}$$

where $I_a \cap I_b$ is the subset of cloud services commonly used by users a and b, $r_{a,i}$ is the normalized QoS value of service 'i' used by user a, and $\tilde{I}(x)$ is an indicator function given as;

$$\tilde{I}(x) = \begin{cases} 1 & if\ x < 0 \\ 0 & otherwise \end{cases}. \tag{4}$$

From the above definition, the ranking similarity between two rankings is in the interval $[-1, 1]$, where -1 is obtained when the order of user a is the exact reverse of user b. Given that KRCC compares service pairs, the intersection between two users has to be at least $2(|I_a \cap I_b| \geq 2)$ for making a similarity computation.

A set of similar users S(a) is identified for the active user a by

$$\mathrm{S}(a) = \{b | b \in \mathrm{Top\_K}(a), \mathrm{SIM}(a,b) > 0, b \neq a\}, \tag{5}$$

where Top_K(a) is a set of the Top_K similar users of active user a. Top_K Similar users are acknowledged by arranging their similarity values in descending order. SIM(a,b) > 0 that excludes the dissimilar users with negative similarity values. In this paper, we make use of the hybrid Top-K algorithm to select neighbors. First, for the capable users, we set a similarity threshold value. Then the outcome of the dissimilar users are removed by adding the threshold value.

In this correlation technique, two or more QoS parameters are combined to find its correlated properties. Here throughput and response time are combined. When the throughput QoS value is high, it is said to be maximum; but for a response time when the value is less, it is said to be maximum. Since both QoS values are contradictory, normalization has to be performed to put both values in a common range (0–1). Here, the response time is manipulated as (1—response time).

Let us consider $S_i(a)$ as a set of similar users for the active user a based on the ith normalized QoS parameter. In our approach, a set of QoS correlated similar users CS(a) is identified for the active user a by

$$CS(a) = S_1(a) \cap S_2(a) \cap \cdots \cap S_n(a) \tag{6}$$

where 'n' represents the number of QoS parameters.

3.2 QoS Data Smoothing

Data smoothing is an important technique that is used to remove noise from a dataset, and allowing important patterns to stand out and improve the accuracy of the QoS prediction. Assume that u_t is one of the similar users of u, and we want to predict the QoS of service 'i' to user u. Traditional methods replace the QoS of service 'i' to u_t with 0 if u_t has not invoked service 'i'. Therefore, this process lowers the accuracy of the predicted QoS. This problem is handled in our paper with the help of the data smoothing concept.

The most preferred two partition-based cluster algorithms, notably K-Means and FCM are well-known. K-Means is one of the trouble-free unsubstantiated learning algorithms to solve clustering problems [54]. Through the result of performance based methods of Velmurugan [54], it is evident that the results are accurate, and easily understandable in FCM compared with K-Means [54]. Hence, in this approach, in order to improve the accuracy of the QoS value prediction, the data smoothing, as described in [42, 51], is done with the help of FCM clustering instead of a K-Means algorithm.

3.3 Correlated QoS Ranking

Given that the user-observed QoS values in a normalized form on two cloud services, the user preference function between these two services can be easily derived by comparing the normalized QoS values, where

$$£(i,j) = q_i - q_j. \tag{7}$$

$£(i,j)$ is the preference function obtained where i and j represent services and q_i and q_j represents the normalized QoS value of the service 'i' and 'j'. In our outlook of reality, our goal is to produce ranking for users, spotlight modelling a user's preference function of the form as on [39, 58], £: I × I → IR, where £(i,j) > 0.

Here ith service is compared with the jth service that is ith service is better than jth service, that means that service i is more preferable to j for present dynamic user a and vice versa. Suppose user a's QoS throughput value on cloud service 'i' and 'j' are 5 and 3, respectively, This clearly indicates that the user prefers cloud service 'i' to the cloud service j as an indication for (i, j) > 0.

The magnitude of this preference function |£(i,j)| indicates the strength of preference and a value of zero means that there is no preference between the two services. Assume that £(i, i) = 0 for all i € I and that £ is anti-symmetric, i.e., £(i, j) = −£(j, i) for all i, j € I.

With the help of the user preference function calculation, the most similarity between services is gained as a £:(N × N) matrix. The result will be M number of users that will have M number of user preference functions such as

$$[£_1, £_2, £_3, £_4, \ldots, £_M], \tag{8}$$

To obtain the preference values regarding pairs of services that have not been raised or used by the present user, the preference values of similar correlated users CS(a) are engaged. Generally, stronger confirmation in priority is given by £(i,j) > 0 for the present user where frequently similar correlated users in CS(a) view service 'i' is of higher quality than service 'j'. This shows the way for manipulating the value of preference function £(i, j), where service 'i' and service 'j' are not explicitly viewed by the present user a,

$$£(\mathrm{i},\mathrm{j}) = \sum\nolimits_{b \in \mathrm{CS}(a)} \mathrm{W}_b(\mathrm{q}_{b,\mathrm{i}} - \mathrm{q}_{b,\mathrm{j}}), \tag{9}$$

where b is a similar correlated user of the present user a, CS(a) is a subset of similar users of a, $q_{b,i}$ and $q_{b,j}$ represent the QoS value of the service i and j accessed by the user b. W_b is a weighting factor of the correlated similar user b, which can be estimated by

$$W_b = \frac{SIM(a,b)}{\sum_{b \in CS(a)} SIM(a,b)}. \tag{10}$$

W_b confirms that a correlated similar user with a higher similarity value has a better impact on the preference value prediction in Eq. (9). In the existing system, only the user b who accessed the both the services 'i' and 'j' is taken as the similar user of a. In our approach, if the correlated user didn't access the service i or j, then the FCM clustering based data smoothing process is done to estimate $q_{b,i}, q_{b,j}$ as described in [26]. Fuzzy C-Means clustering is used for clustering purposes on user similarity function. We assume all users into k group clusters as U = $\{u_1, u_2, \ldots, u_n\}$, clustering results are represented as $\{C_u^1, C_u^2, \ldots, C_u^k\}$. Given b belongs to cluster C_u i.e., $C_u \in \{C_u^1, C_u^2, \ldots, C_u^k\}$, QoS vector $q_{b,i}$ is given as;

$$q_{b,i} = \tilde{q}_b + \Delta rC_u(i), \tag{11}$$

where $\Delta rC_u(i)$ is the average QoS derivations of service 'i' to every users in cluster C_u

$$\Delta rC_u(i) = \frac{\sum_{u' \in C_{u(i)}} (q_{u',i} - q'_u)}{|C_u(i)|} \tag{12}$$

where $C_u(i) \in C_u$ is the set of clusters C_u who have invoked service 'i' and $|C_u(i)|$ is the cardinality of $C_u(i)$. After clustering based on user similarity average QoS for particular service has been employed.

Algorithm: CorQoSCloudRank

Input: an Accessed service set [AS_1, AS_2, ...AS_m], a Full service set FS, a Summation of Normalized Preference function [$S£_1$, $S£_2$, ...$S£_m$], a Summation of QoS values [SNQ_1, SNQ_2, ...SNQ_m], an active user *a*

Output: a service ranking ℟ for an active user *a*

```
1  W = AS_a;
2  ernk =1;
3  while W ≠ ϕ do
4  |  m = arg max_i ∈ SNR_a^i
5  |  Ė℟ = ernk;
6  |  ernk = ernk + 1;
7  |  W = W - {m};
8  end
9  foreach i ∈ FS do
10 |  Π_a(i) = ∑_{j∈FS} S£_a (i, j);
11 end
12 rnk = 1;
13 while FS ≠ ϕ do
14 |  m = arg max_i Π_a(i);
15 |  if m is accessed by user a
16 |  |  call updaternk1(℟, rnk, S£_a, Π_a, FS, m);
17 |  else
18 |  |  nrnk = ϕ;
19 |  |  foreach u ∈ CS(a) do
20 |  |  |  if m is accessed by user u
21 |  |  |  |  foreach i ∈ FS do
22 |  |  |  |  |  Π_u(i) = ∑_{j∈SFS_u} S£_u(i,j);
23 |  |  |  |  end
24 |  |  |  |  SL = sort(Π_u, 'descend');
25 |  |  |  |  rnk_um = index of m (SL);
26 |  |  |  |  nrnk = nrnk ∪ { rnk_um};
27 |  |  |  end
28 |  |  end
29 |  |  if sizeof(nrnk) == 0
30 |  |  |  call updaternk(℟, rnk, S£_a, Π_a, FS ,m);
31 |  |  end
32 |  |  if sizeof(nrnk) == 1
33 |  |  |  if absdiff(nrnk(1), rnk) ≤ 1
34 |  |  |  |  updaternk2(℟, nrnk(1), S£_a,Π_a, FS ,m);
35 |  |  |  else
36 |  |  |  |  updaternk1(℟, rnk, S£_a,Π_a, FS ,m);
37 |  |  |  end
38 |  |  end
39 |  |  if sizeof(nrnk) > 1
40 |  |  |  mfrnk = -1;
41 |  |  |  ncnt = 1;
42 |  |  |  foreach p ∈ nrnk do
43 |  |  |  |  if count_p(nrnk) > ncnt
44 |  |  |  |  |  ncnt = count_p(nrnk);
45 |  |  |  |  |  mfrnk = p;
46 |  |  |  |  end
47 |  |  |  end
```

```
48        if mfrnk != -1
49          if absdiff (mfrnk, rnk) <= 1
50            call updaternk2(ℜ, mfrnk, S£_a, Π_a, FS ,m);
51          else
52            call updaternk1(ℜ, rnk, S£_a, Π_a, FS ,m);
53          end
54        else
55          avgrnk = floor(mean(nrnk));
56          if absdiff(avgrnk, rnk)<=1
57            call updaternk2(ℜ, avgrnk, S£_a, Π_a,  FS ,m);
58          else
59            call updaternk1(ℜ, rnk, S£_a, Π_a, FS ,m);
60          end
61        end
62      end
63    end
64  end
65  while AS_a ≠ ϕ do
66    e  = arg min_{i∈AS_a} ℜ̂i;
67    index = min_{i∈AS_a} ℜ(i);
68    ℜ(e) = index;
69    AS_a = AS_a - {e};
70  end

71  Procedure updaternk1(ℜ, rnk, S£_a, Π_a, FS ,m)
72    global  ℜ, rnk, S£_a, Π_a, FS ,m;
73      while rnk ≠ ℜ do
74        rnk = rnk +1;
75      end
76      ℜ(m) = rnk;
77      rnk = rnk +1;
78      FS = FS - {m};
79      foreach i ∈ FS do
80        Π_a(i) = Π_a(i) -  S£_a(i,m);
81      end
82  End Procedure

83  Procedure updaternk2(ℜ, urnk, S£_a, Π_a, FS ,m)
84    global ℜ, S£_a, Π_a, FS ,m;
85    foreach i ∈ ℜ do
86      if ℜ(i) ≥ urnk
87        ℜ(i) = ℜ(i) + 1;
88      end
89      ℜ(m) = urnk;
90      FS = FS - {m};
91      foreach i ∈ FS do
92        Π_a(i) = Π_a(i) - S£_a(i, m);
93      end
94    end
95  End Procedure
```

3.4 CorQoSCloudRank

Qiu et al. [57] proposed a reputation-aware QoS value prediction approach that first calculates the reputation of each user based on their contributed values, and then takes advantage of its reputation-based ranking to exclude the values contributed by untrustworthy users. Wu et al. [41] presented a ranking method called ServiceRank that considers QoS aspects, such as response time and availability, as well as the social perspectives of services.

In collaborative filtering, the ranking on services is estimated on the basis of correlation, among the entire users' service nature. Each user has his/her own ranking level for their preferred services according to the past QoS ranking system. For an 'Employed' service set, ranking is manipulated with the help of preference function calculation. For an 'Unemployed' service set, ranking is manipulated by picking the correlated neighbors who have chosen the service. For this algorithm, the following values are taken as input: the full service set (FS), the accessed service set for each user in M is (AS_1, AS_2,…,AS_m), added normalized QoS values of each user (SNQ_1, SNQ_2,…,SNQ_n) and summation of preference function on each QoS values for all users in '*m*'.

CorQoSCloudRank algorithm explanation:

- *Step 1(lines 1–8).* Rank the accessed cloud services in W based on the summation of normalised observed QoS values. ĘR stores the ranks of the accessed service, where ĘR(*m*) returns the rank of the service *m*, where *m* is a necessary cloud service that is accessed by the active user *a*.

From this procedure, rank is achieved for the accessed services. The Fig. 2 clearly depicts that an active user *a* provides rank for the accessed services s1 to sn, but there is no rank for the service s2 because it is not accessed. The following procedure has to be followed to find rank level for all the services, which are not accessed by the active user *a*.

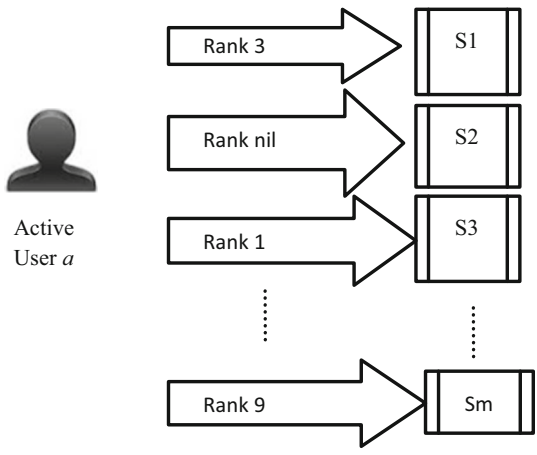


Fig. 2 Active user a ranking services

- *Step 2(lines 9–11).* For each service in the full service set FS, the sum of the summation of preference values with all other services was calculated by $\Pi_{\mathrm{a}}(\mathrm{i}) = \sum_{j \in \mathrm{FS}} \mathrm{S£_a}(i,j)$, where a is an active user. The larger $\Pi_{\mathrm{a}}(\mathrm{i})$ value indicates active user *a* prefers ith service more than the other services.
- *Step 3(lines 12–70).* Here, the Correlated ranking scheme is applied to find the rank level for each service in FS. The line no 14 is executed to find the service *m* that has the maximum $\Pi_{\mathrm{a}}(m)$ value. When the mth service is accessed by active user *a* then it will call the updaternk1 procedure i.e., (lines 71–82). In that procedure, the accessed service *m* is assigned with rank *rnk*, following the lines (65–70).

After that, as per the procedure from lines (78–81) the selected accessed service *m* is then removed from the full service set FS. The preference function values $\Pi_a(\mathrm{i})$ of the remaining services are updated to remove the effects of the selected accessed service *m*.

If *m*th service is not accessed by active user *a*, then (lines 20–28) will be executed. Here we check whether the correlated neighbors have chosen the *m*th service, if they are chosen, their ranks(*nrnk*) for *m*th service are retrieved with the help of their preference function. Three possibilities are available for choosing the rank on the basis of correlated ranks(*nrnk*) provided by correlated neighbors (Fig. 3).

In this figure, the 2nd service is allotted with rank 3, because the correlated neighbors have not chosen the service and active user *a* has afforded rank 3.

- *Step 4 (lines 29–31).* First possibility is that the *nrnk* set is empty i.e., no one in the correlated neighbors chose the service *m*. In this case, the corresponding updaternk1 procedure is called to assign the rank with *rnk*.
- *Step 5 (lines 32–38).* In the second possibility, only one correlated neighbor provides rank *nrnk*(1) for the service *m*. Then line 33 checks if the rank value *rnk* (estimated by the active user on the basis of the preference function) differs

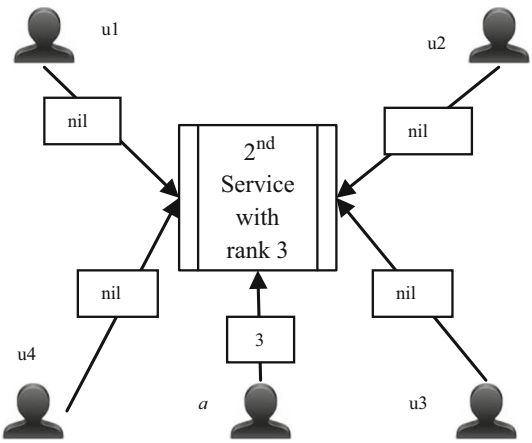


Fig. 3 First possibility

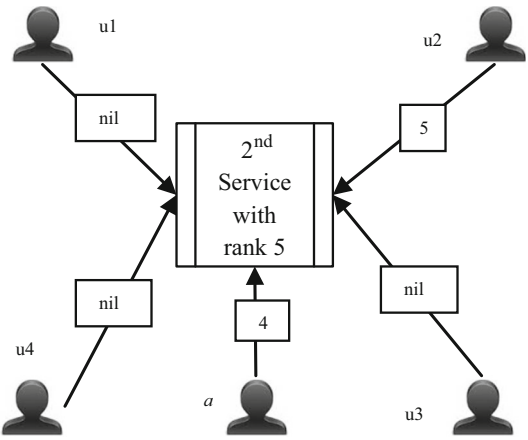


Fig. 4 Second possibility with single rank

more than one position level compared to the *nrnk*(1) means, if so it will call the updaternk1 procedure from lines (71–82); otherwise, it will call the updaternk2 procedure from lines (83–95) (Fig. 4).

In this figure, the 2nd service is allotted with rank 5, because the rank provided by user 2 is the 5th rank and the rank estimated by active user a is the 4th rank. Both ranks differ only in one position level, but this is just opposite in the example shown in Fig. 5.

In this figure, the 2nd service is allotted with rank 7, because the rank provided by the user 2 is the 3rd rank and the rank estimated by active user *a* is the 7th rank, so both ranks do not satisfy the condition. So the 7th rank is assigned for the 3rd service.

In the updaternk2 procedure, the service *m* is assigned with rank *nrnk*(1), before it processes the services that are already allotted with the rank as *nrnk*(1) and the above will be incremented by one level.

After that, as per the procedure from (lines 84–87) the selected service *m* is then removed from the full service set FS. The preference function values $\Pi_a(i)$ of the remaining services are updated to remove the effects of the selected service *m*.

- *Step 6(lines 39–63).* In the third possibility, as shown in Figs. 6 and 7, more than one correlated neighbor provides rank for the service that is not accessed by the active user *a*. In this case, as per the lines (40–47), it will find the most frequent rank (mfrnk) from the correlated ranks (*nrnk*). Then, as per line number 48, it checks the availability of the most frequent rank, and if it so available, the most frequent rank(mfrnk) will update the rank; otherwise, it will choose the average of the correlated neighbors ranks(avgrnk).
- In this figure, correlated users u2 and u4 have allotted the rank as 2, and since rank 2 has occurred frequently, we have considered the frequent rank as 2.
- In both cases, it will check line numbers 49 and 56, to see whether the chosen rank (avgrnk/mfrnk) differs more than one position level compared to rank *rnk*

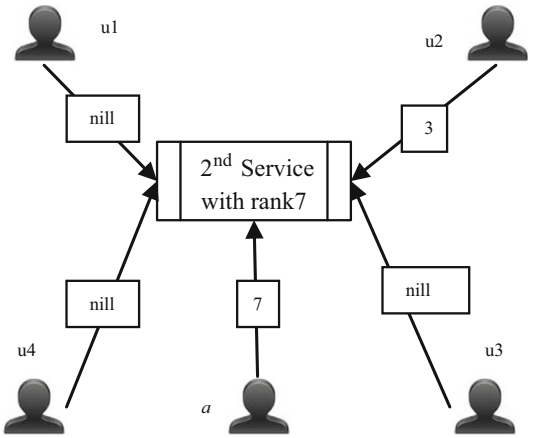


Fig. 5 Second possibility with update rank

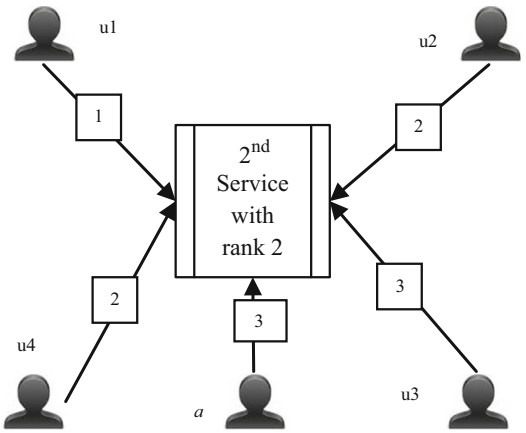


Fig. 6 Third possibility with frequent rank

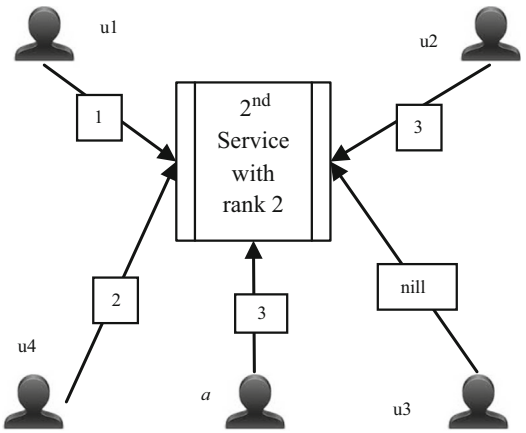


Fig. 7 Third possibility with average rank

which was estimated by the active user *a*. If the condition is satisfied call updaternk1 procedure or it will call updaternk2 procedure.

This figure clearly shows that there is no frequent rank available, so with the help of the correlated neighbors rank, we can find the average rank.

- *Step 7 (lines 65–70).* In this part, the ranks of the accessed services in $\mathbb{R}$ is corrected with the help of $\hat{E}\mathbb{R}$.

4 Experiments

4.1 Dataset Description

To evaluate the correlated QoS ranking accuracy, we used the detailed WSDream-QoS dataset values that were publicly released online by Zheng et al. [60]. This dataset consists of QoS values of the 500 real-world web services viewed by the 300 service users. It is represented as a 300 $\times$ 500 user-item matrix, where each item in the matrix is the QoS value of a web service observed by a user. Totally 150,000 web service invocations are provided. The response time and throughput values of each invocation are given. Our experiment has been carried out with MATLAB 13. In our experiment, the QoS values are employed to rank the services that are to be correlated.

4.2 Evaluation Metric

The QoS ranking prediction is to predict QoS values as accurate as possible. If the QoS numerical values are given as class or labels our algorithm provides good results for qualitative variables also. In order to evaluate the ranking prediction accuracy, we employ the Normalized Discounted Cumulative Gain (NDCG) [58, 61] metric, which is a popular metric for evaluating ranking results. The NDCG measures the performance of a recommendation system based on the condition significance of the recommended entities. It varies from 0.0 to 1.0, with 1.0 representing the ideal ranking of the entities. Given an ideal service QoS ranking (used as ground truth) and a correlated QoS ranking, the NDCG value of the Top-K ranked services can be estimated by

$$NDCG_k = \frac{DCG_k}{IDCG_k}, \tag{13}$$

where DCG_k and $IDCG_k$ are the Discounted Cumulative Gain (DCG) values of the Top-K services of the correlated ranking and ideal ranking. The value of DCG_k can be estimated by

$$\mathrm{DCG_k} = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i}. \quad (14)$$

where $\mathrm{rel_i}$ is the graded relevance QoS value of the service at position i of the ranking.

The premise of DCG is that a high-quality web service appearing lower in the ranking list should be punished as the graded relevance value that is reduced as logarithmically proportional to the position of the result. The DCG value is gathered cumulatively from the top of the result list to the bottom with the gain of each result discounted at lower ranks. The ideal rank achieves the highest gain among all different rankings. The $\mathrm{NDCG_k}$ value is on the interval of 0–1, where a larger value stands for a better ranking accuracy, indicating that the correlated ranking is closer to the ideal ranking. The value of 'k' is in the interval of 1 to *m*, where *m* is the total number of cloud services.

4.3 Performance Comparison

The analysis on ranking prediction is done through six different techniques of correlated QoS ranking scheme. In the former three models of correlated QoS ranking, the correlation properties of multiple QoS parameters are used in the user similarity estimation process. The later three methods utilize the correlation property for user similarity as well as ranking prediction. The six proposed correlated QoS ranking algorithms are as follows,

1. QoS Correlated User based CloudRank without Data Smoothing (QCUCR).
   - This method employs correlation property among multiple QoS values only for user similarity estimation. The preference function value of each QoS parameter is added together to predict the rank on the basis of ideal ranking in descending order.

2. QoS Correlated User based CloudRank with (Data Smoothing) K-Means (QCUCRK).
   - This method is similar to QCUCR. In this approach, a pre-processing technique, such as data smoothing is employed with the help of the K-Means algorithm.

3. QoS Correlated User based CloudRank with (Data Smoothing) FCM (QCUCRFC).
   - This method is similar to QCUCR. In this approach, a pre-processing technique, such as data smoothing is employed with the help of the FCM algorithm.

Table 2 Analysis on NDCG versus manifest using PCC

Methods	Matrix density = 10 %			Matrix density = 30 %			Matrix density = 50 %		
	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100
QCUCR	0.5447	0.6607	0.6708	0.6032	0.6764	0.7305	0.7271	0.7103	0.7016
QCUCRK	0.5522	0.6802	0.6742	0.6244	0.6863	0.7356	0.7324	0.7123	0.7063
QCUCRFC	0.5641	0.6932	0.6843	0.6414	0.6893	0.7377	0.7369	0.7177	0.7103
CQCR	0.5702	0.7047	0.7083	0.6766	0.7073	0.7502	0.7489	0.7363	0.7286
CQCRK	0.5794	0.7105	0.7143	0.6804	0.72	0.7589	0.7612	0.7516	0.7354
CQCRFC	0.5845	0.7177	0.7186	0.6937	0.7253	0.7756	0.7655	0.7585	0.7584

Table 3 Analysis on NDCG versus Manifest using SRCC

Methods	Matrix density = 10 %			Matrix density = 30 %			Matrix density = 50 %		
	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100
QCUCR	0.5642	0.6937	0.7308	0.7175	0.7316	0.8023	0.7917	0.7922	0.8417
QCUCRK	0.5683	0.7069	0.7373	0.7217	0.7382	0.8099	0.8002	0.8062	0.8452
QCUCRFC	0.5714	0.7138	0.7403	0.7264	0.7423	0.8122	0.8075	0.8212	0.8469
CQCR	0.5784	0.7186	0.7464	0.7318	0.7537	0.8202	0.8158	0.8256	0.8558
CQCRK	0.5803	0.7205	0.7481	0.7362	0.7631	0.8238	0.8191	0.8318	0.8601
CQCRFC	0.5872	0.7273	0.7525	0.7389	0.7673	0.8275	0.8244	0.8364	0.8633

Table 4 Analysis on NDCG versus manifest using KRCC

Methods	Matrix density = 10 %			Matrix density = 30 %			Matrix density = 50 %		
	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100	NDCG1	NDCG10	NDCG100
QCUCR	0.5874	0.7152	0.7598	0.7423	0.7667	0.8414	0.8179	0.8482	0.8949
QCUCRK	0.5893	0.7187	0.7623	0.7468	0.7704	0.8511	0.8267	0.8496	0.8965
QCUCRFC	0.5905	0.7201	0.7687	0.7514	0.7767	0.8526	0.8286	0.8512	0.8973
CQCR	0.5923	0.7232	0.7707	0.7578	0.7849	0.8584	0.8395	0.8581	0.8985
CQCRK	0.6007	0.7297	0.7756	0.7582	0.7906	0.8602	0.8416	0.8632	0.8994
CQCRFC	0.6053	0.7368	0.7792	0.7599	0.7953	0.8689	0.8484	0.8697	0.9091

4. Correlated QoS CloudRank without Data Smoothing (CQCR).
   - This method employs correlation property among multiple QoS values for user similarity estimation as well as to predict personalized ranking.

5. Correlated QoS CloudRank with (Data Smoothing) K-Means (CQCRK).
   - This method is similar to CQCR. In this approach, a pre-processing technique such as data smoothing is employed with the help of the K-Means algorithm.

6. Correlated QoS CloudRank with (Data Smoothing) FCM (CQCRFC).
   - This method is similar to CQCR. In this approach, a pre-processing technique, such as data smoothing is employed with the help of the FCM algorithm.

In the actual world, the user-item matrixes are generally very sparse since a user normally only chooses a very rare number of cloud services. So to make our experiments practically, we randomly remove entries from the user-item matrix to make sparser with various densities. The user-item matrix density (i.e., proportion of nonzero entries) is reduced randomly to d %.

In our experiment, all the six proposed correlated QoS ranking algorithms employ a matrix density from 10 to 50 % percent with the step value as 5 %. While evaluating the ranking accuracy, each six proposed correlated QoS ranking algorithms are executed up to 30 times and the average value is illustrated in this paper. The rankings based on the unique full matrix are utilized as model rankings to learn the QoS ranking accuracy. The Top-K value is set to 10 in the prediction process. The threshold assigned for the hybrid Top-K algorithm is 0.25.

The three traditional similarity computation methods such as the PCC, SRCC and KRCC are employed for all six proposed correlated QoS ranking algorithms, whose performance is evaluated in the form of NDCG1, NDCG10 and NDCGG100.

Tables 2, 3 and 4 shows the performance analysis of the NDCG based correlated cloud rank approach with the PCC, SRCC and KRCC calculated for 0, 30, and 50 % density for user-item matrix (Figs. 8, 9, 10).

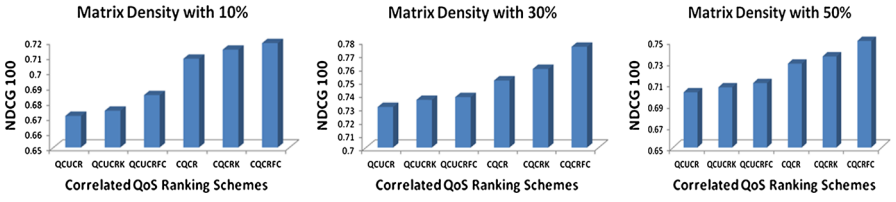


Fig. 8 Impact of PCC with correlation ranking scheme

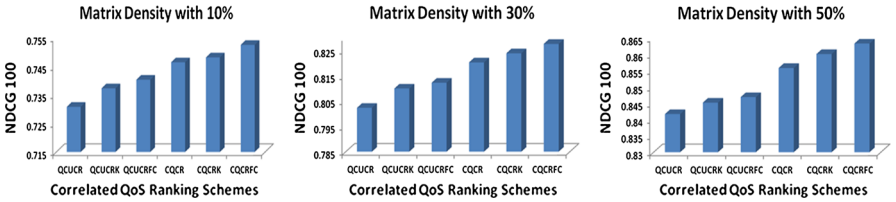


Fig. 9 Impact of SRCC with correlation ranking scheme

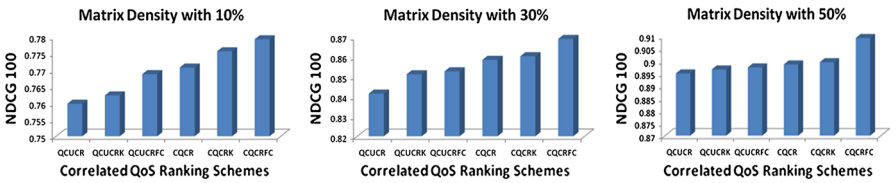


Fig. 10 Impact of KRCC with correlation ranking scheme

The result analysis shown in Tables 2, 3 and 4 clearly describes that the ranking accuracy (in terms of NDCG values) increases as the density of the user-item matrix also increases from 10 to 50 %. The denser user-item matrix provides more data for the ranking accuracy.

Figure 11 clearly shows that the ranking accuracy has been improved as the density of user-item matrix has been increased from 10 to 50 %. In our work, the CQCR-PCC in 50 % outperforms CQCR-PCC in 10 % with 0.0203 NDCG same for SRCC and KRCC with 0.1094 NDCG and KRCC with 0.1278 NDCG.

The inclusions of the pre-processing technique, i.e., the data smoothing technique enhance the accuracy level. Compared with the K-Means data smoothing algorithm technique, the FCM algorithm based data smoothing consistently achieves a better ranking accuracy.

Figure 12 clearly shows that CQCRFC outperforms CQCR with 0.0298 NDCG for PCC, with 0.0075 NDCG for SRCC, with 0.0106 NDCG for KRCC and

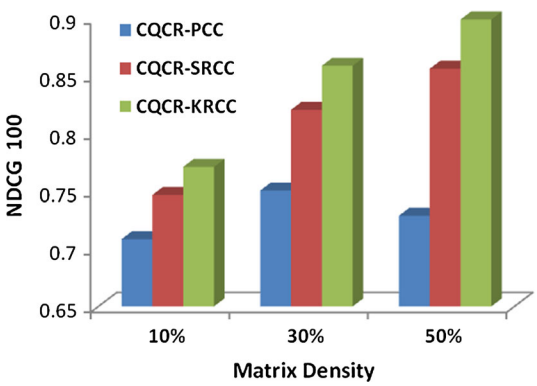


Fig. 11 Matrix density 10, 30 and 50 %

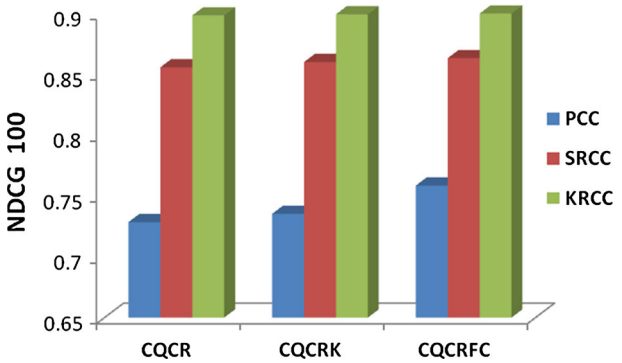


Fig. 12 Impact of data smoothing

outperforms CQCRK with 0.023 NDCG for PCC, with 0.0032 NDCG for SRCC, with 0.0097 NDCG for KRCC. This examination indicates that, the correlation property in the combined form of QoS with the data smoothing technique improves the ranking accuracy for multiple QoS constraints.

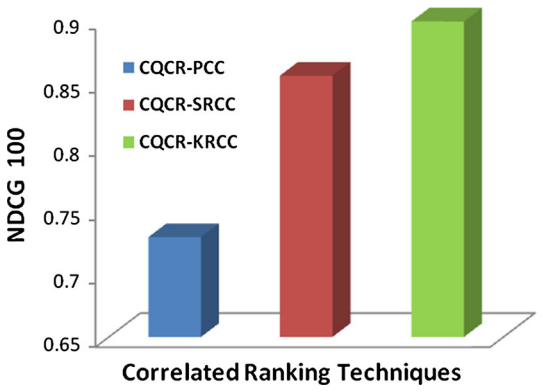
On the basis of the similarity computation technique, the KRCC method obtained improved results compared to the PCC and SRCC.

Figure 13 clearly shows that the algorithm with the KRCC outperforms SRCC by 0.0427 NDCG and outperforms the PCC by 0.1699 NDCG.

While employing the correlation property of multiple QoS values for similarity estimation and ranking prediction yields a better result than applying the correlation property only for a similarity estimation.

Figure 14 clearly shows that the Correlated QoS CloudRank without Data Smoothing (CQCR) outperforms the QoS Correlated User based CloudRank without Data Smoothing (QCUCR). CQCR outperforms with 0.027 NDCG for PCC, with 0.0141 NDCG for SRCC and with 0.0036 NDCG for KRCC when compared with QCUCR.

Fig. 13 Impact of similarity computation methods



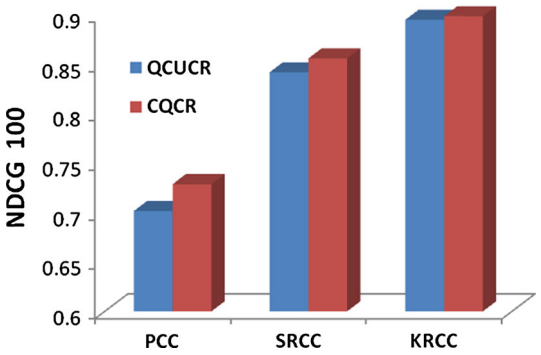


Fig. 14 Impact of QoS correlation property in ranking

The Correlated QoS CloudRank with (Data Smoothing) FCM (CQCRFC) technique obtains the improved prediction accuracy (largest NDCG values) with combined QoS under all the experimental settings consistently. It achieves a higher ranking accuracy than the other five techniques. Figure 15 clearly shows that the KRCC with FCM outperforms CQCR with 0.0106 NDCG and outperforms CQCRK with 0.0097 NDCG.

In [58], the CloudRank2 (response time) approach with its ideal ranking (single QoS) it achieves a ranking accuracy 0.8884 NDCG for 50 % matrix density and KRCC for similarity estimation, for CloudRank2 (throughput) it is 0.8943 NDCG. The proposed approach CQCRFC compared with its ideal ranking (multiple QoS) it achieves 0.9091 NDCG for 50 % matrix density.

Figure 16 clearly shows that, the CQCRFC technique improves the ranking accuracy with 0.0148 NDCG compared with CloudRank2 (throughput) and with 0.0207 NDCG CloudRank2 (response time).

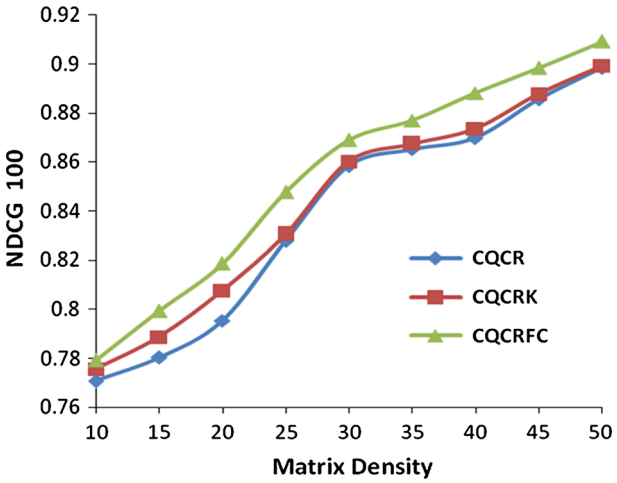


Fig. 15 Impact of Kendall with FCM

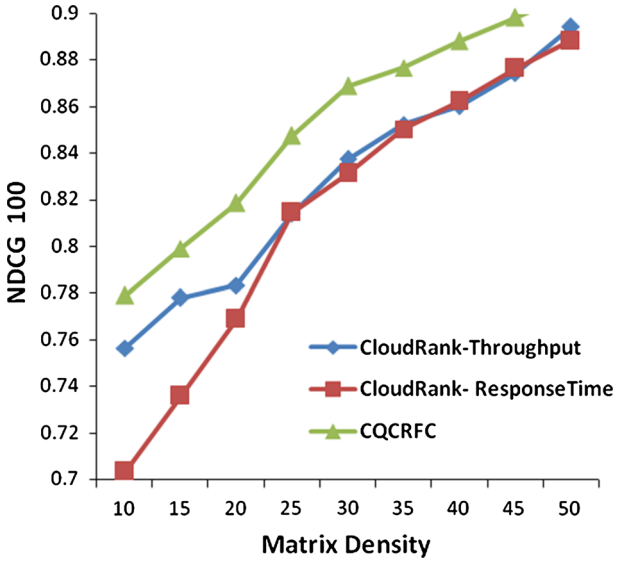


Fig. 16 Impact of correlated property of multiple QoS

5 Related work

Cloud computing is gaining popularity these days. There are numerous works on cloud computing such as multimedia communication [13], virtualization [38, 46, 50], load balancing [10], fault tolerance [24, 40], service pricing [55], profit maximization [2, 4], admission control [32], service composition [6], queuing systems [3, 8], and resource monitoring [56]. The QoS is also an important concept in cloud computing [5, 34, 41]. Different applications have different QoS requirements [5, 15, 57]. QoS of cloud services can be compared either from the client side or at the server side (e.g.; price, availability, etc.). QoS measures have been used for various approaches such as resource management [11, 15], resource scheduling [9, 20, 23, 30, 48], service measurement [45], data replication [25], swarm optimization [14], and resource optimization [49]. This paper focuses on predicting optimal service selection using Correlated QoS ranking to improve accuracy.

Recommendations are specified to the user based on assessment of items by other users from the same group, with whom he/she shares common preferences. If the item has been positively rated by the community, it will be recommended to the user. Collaborative filtering methods are widely adopted in recommender [28] and QoS systems [17]. Two types of collaborative filtering approaches are widely studied memory based [21, 22, 27, 43] and model based [21, 22, 27, 37, 43]. The model-based approaches group together different users in the training database into a small number of classes based on their rating patterns. In order to predict the rating from a test user on a particular item, these approaches first categorize the test user into one of the predefined user classes and use the rating of the predicted class on

the targeted item as the prediction [37, 43]. Algorithms within this category include bayesian network approaches [27], the aspect model [33, 52], gradient descent [12] and the latent class models [52, 53].

Two types of memory-based methods [29] have been studied: user-based [21, 43] and item-based [7, 21, 37, 39]. User-based methods first look for similar users who have similar rating styles with the active user and then employ the ratings from those similar users to predict the ratings for the active user. Item-based methods share the same idea with user-based methods. The only difference is user-based methods to find the similar users for an active user but item-based methods try to find the similar items for each item. However, the most common method used in a collaborative filtering method is the user based model [7, 43]. To find similarity between users, the Vector Similarity (VC) [16, 39], Cosine Based Similarity [7, 36, 37], KRCC [39, 58], SRCC and PCC [21, 22, 43], methods were employed. The present work is different from the preceding work [57, 58, 59] in the sense that the accuracy of QoS based ranking has improved with the help of correlation and combination of QoS properties. Still enormous effort is needed for employing collaborative filtering methods for Web service QoS value prediction.

6 Conclusion and Future Work

In this paper, we propose a correlated QoS ranking algorithm to predict personalized ranking for service selection for an active user. Six different kinds of correlated ranking algorithms were proposed and we compared their accuracy in terms of NDCG. The process of selecting similar neighbors for an active user is a very important one for the accuracy of prediction; hence, in this scheme we proposed a QoS correlation property based user selection with hybrid Top-K algorithm. Multiple QoS correlation properties were efficiently extracted and combined properly to predict the rank for the cloud service. The investigational results show that our approach improve the accuracy of the QoS ranking prediction.

For future work, we would like to investigate time-aware correlated QoS ranking approaches for cloud services by using data collected from service users, cloud services, and time. Apart from the correlation property of QoS, the location aware ranking prediction scheme can be extended in the future. Furthermore, we also plan to detect and handle malicious QoS values provided by users.

Acknowledgments The authors would like to thank Zibin Zheng, Yilei Zhang and Michael R. Lyu for providing the WSDream-QoS datasets [60] that were publicly released from the website (http://www.wsdream.net). This dataset was very helpful for our research purposes. We would also like to thank the anonymous reviewers for their valuable and insightful suggestions.

References

1. Amazon.: Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/,1 (2009)
2. Ani Brown Mary, N.: Profit maximization for SAAS using SLA based SPOT PRICING in CLOUD COMPUTING. Int. J. Emerg. Technol. Adv. Eng. **3**(1), 19–25 (2013)

3. Ani Brown Mary, N., Saravanan, K.: Performance factors of CLOUD COMPUTING data centers using [(M/G/1):(/GDMODEL)] queuing systems. Int. J. Grid Comput. Appl. **4**(1), 1–9 (2013)
4. Ani Brown Mary, N.: Profit maximization for service providers using hybrid pricing in cloud computing. Int. J. Comput. Appl. Technol. Res. **2**(3), 218–223 (2013)
5. Ani Brown Mary, N., Jayapriya, K.: An extensive survey on QoS in cloud computing. Int. J. Comput. Sci. Inf. Technol. **5**(1), 1–5 (2014)
6. Al Falasi, A., Serhani, M.A.: A framework for SLA-based cloud services verification and composition. In: Proceedings of 2011 International Conference on Innovations in Information Technology (2011)
7. Sarwar, B., Karypis, G., Konstan, J. & Riedl J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of WWW Conference (2001)
8. Bharathi, M., Sandeep Kumar, P., Poornima, G.V.: Performance factors of cloud computing data centers using M/G/m/m+r queuing systems. IOSR J. Eng. **2**(9), 06–10. e-ISSN: 2250-3021, p-ISSN: 2278-8719. www.iosrjen.org (2012)
9. Li, B., Song, A.M., Song, J.: A distributed QoS-constraint task scheduling scheme in cloud computing environment: model and algorithm. Adv. Inf. Sci. Serv. Sci. **4**(5), 283–291 (2012)
10. Mondala, B., Dasguptaa, K., Duttab, P.: Load balancing in cloud computing using stochastic hill climbing—a soft computing approach. Proced. Technol. **4**, 783–789 (2012)
11. Yeo, C.S. Buyya, R.: A taxonomy of market-based resource management systems for utility-driven cluster computing. Softw. Pract. Exp. **36**, 1381–1419 (2006). Published online 8 June 2006 in Wiley InterScience (www.interscience.wiley.com). doi:10.1002/spe.725
12. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of ICML 2005, pp. 89–96 (2005)
13. Angeli, D., Masala, E.: A cost-effective cloud computing framework for accelerating multimedia communication simulations. J. Parallel Distrib. Comput. **72**(10), 1373–1385 (2012)
14. Kumar, T.A.D., Sumathi, G.: Intelligent management of remote facilities and quality of cloud services. Int. J. Grid Distrib. Comput. **4**(2), 43–51 (2011)
15. Armstrong, D., Djemame, K.: Towards quality of service in the cloud. In: Proceedings of the School of Computing, University of Leeds, United Kingdom
16. Wu, D., Mendel, J.M.: A vector similarity measure for linguistic approximation: interval type-2 and type-1 fuzzy sets. Inf. Sci. **178**, 381–402 (2008)
17. Adomavicius, G., Kwon, Y.O.: Improving aggregate recommendation diversity using ranking-based techniques. IEEE Trans. Knowl. Data Eng. **24**(5), 896–911 (2012)
18. Google, App Engine. http://code.google.com/appengine/. 17 February 2009
19. Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**, 76–80 (2003)
20. Liu, G., Liu, C., Yang, C. Li, D.: Scheduling research based on genetic algorithm and QoS constraints of cloud computing resources. J. Theor. Appl. Inf. Technol. **51**(1), 91–96 (2013)
21. Xue, G.R., Lin, C., Yang, Q., Xi, W., Zeng, H.J., Yu, Y. Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: Proceedings of SIGIR (2005)
22. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: 30th International ACM SIGIR Conference Research and Development in Information Retrieval (SIGIR'07), pp. 39–46 (2007)
23. Lawrance, H., Silas, S.: Efficient Qos based resource scheduling using PAPRIKA method for cloud computing. Int. J. Eng. Sci. Technol. **5**(3), 638–643 (2013)
24. Dean, J. Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the OSDI 2004
25. Lin, J.W., Chen, C.H., Chang, J.M.: QoS-aware data replication for data intensive applications in cloud computing systems. IEEE Trans. Cloud Comput. **1**(1), 101–115 (2013)
26. Wu, J., Chen, L., Feng, Y., Zheng, Z., Zhou, M.C., Wu, Z.: Predicting quality of service for selection by neighborhood-based collaborative filtering. IEEE Trans. Syst. Man Cybern. Syst. **43**(2), 428–439 (2013)
27. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, Madison (1998)
28. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington, DC (2003)

29. Wang, J., de Vries, A.P., Reinders, M.J.T.: Unifying userbased and itembased collaborative filtering approaches by similarity fusion. In: Proceedings of the *SIGIR'06,* Seattle, Washington, USA, August 6–11 (2006)
30. Kim, Kyong Hoon, Lee, Wan Yeon, Kim, Jong, Buyya, Rajkumar: SLA-based scheduling of bag-of-tasks applications on power-aware cluster systems. IEICE Trans. Inf. Syst. **E93-D**(12), 3194–3201 (2010)
31. L.S.V. Singh, J.A.: A greedy algorithm for task scheduling and resource allocation problems in cloud computing. Int. J. Res. Dev. Technol. Manag. Sci. Kailash **21**(1), (2014). ISBN: 978-1-63102-445-0
32. Wu, L., Garg, S.K., Buyya, R.: SLA-based admission control for a software-as-a-service provider in cloud computing environments. J. Comput. Syst. Sci. **78**, 1280–1299 (2012)
33. Si, L., Jin, R.: Flexible mixture model for collaborative filtering. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC (2003)
34. Devgan, M., Dhindsa, K.S.: A study of different QoS management techniques in cloud computing. Int. J. Soft Comput. Eng. **3**(3), (2013). ISSN: 2231-2307
35. Dodge, M.: Finding the source of the Amazon.com: hype of the "EARTH'S biggest bookstore. In: Proceedings of the Centre for Advanced Spatial Analysis Working Paper Series
36. Khatr, M.: Cosine similarity function for the temporal dynamic web data. Int. J. Comput. Sci. Eng. Technol. **3**(8), 315–318 (2012)
37. Deshpande, M., Karypis, G.: Item-based top-n recommendation. ACM Trans. Inf. Syst. **22**(1), 143–177 (2004)
38. Sultan, N.: Cloud computing for education: A new dawn? Int. J. Inf. Manag. **30**(2), 109–116 (2010)
39. Liu, N.N., Yang, Q.: EigenRank: a ranking-oriented approach to collaborative filtering. In: Proceedings of the *SIGIR'08, New York*, USA
40. Garraghan, P., Townend, P., Xu, J.: Real-time fault-tolerance in federated cloud environments. In: Proceedings of the 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops
41. Wu, Q., Iyengar, A., Subramanian, R., Rouvellou, I., Silva-Lepe, I., Mikalsen, T.: Combining quality of service and social information for ranking services. In: IBM T.J. Watson Research Center, Skyline Drive, Hawthorne, NY 10532, USA
42. Cannon, R.L., Dave, J.V., Bezdek, J.C.: Efficient implementation of the fuzzy-c-means clustering algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **8**(2), 248–255 (1986)
43. Jin, R., Chai, J.Y., Si, L.: An automatic weighting scheme for collaborative filtering. In: Proceedings of SIGIR (2004)
44. Salesforce.com. CRM salesforce.com. http://www.salesforce.com/
45. Garg, S.K., Versteeg, S., Buyya, R.: SMICloud: a framework for comparing and ranking cloud services. In: Proceedings of 2011 Fourth IEEE International Conference on Utility and Cloud Computing
46. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing—the business perspective. Decis. Support Syst. **51**(1), 176–189 (2011)
47. Sforce: the client/service application development utility. www.salesforce.com
48. Dubey, S., Agrawal, S.: QoS driven task scheduling in cloud computing. Int. J. Comput. Appl. Technol. Res. **2**(5), 595–600 (2013)
49. Ferretti, S., Ghini, V., Panzieri, F., Pellegrini, M., Turrini, E.: QoS-aware clouds. In: 2010 IEEE 3rd International Conference on Cloud Computing
50. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. J. Netw. Comput. Appl. **34**, 1–11 (2011)
51. Chattopadhyay, S.: A comparative study of fuzzy-c-means algorithm and entropy-based fuzzy clustering algorithms. Comput. Inf. **30**, 701–720 (2011)
52. Hofmann, T., Puzicha, J.: Latent class models for collaborative filtering. In: IJCAI, pp. 688–693 (1999)
53. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: Proceedings of the SIGIR'03, Toronto, Canada, July 28–August 1 (2003)
54. Velmurugan, T.: Performance based analysis between K-Means and Fuzzy-C-Means clustering algorithms for connection oriented telecommunication data. Appl. Soft Comput. **19**, 134–146 (2014)
55. Kantere, V., Dash, D., Francois, G., Kyriakopoulou, S., Ailamaki, A.: Optimal service pricing for a cloud cache. IEEE Trans. Knowl. Data Eng. **23**(9), 1345–1358 (2011)

56. Emeakaroha, V.C., Netto, M.A.S., Calheiros, R.N., Brandic, I., Buyya, R., De Rose, C.A.F.: Towards autonomic detection of SLA violations in Cloud infrastructures. Future Gener. Comput. Syst. **28**(7), 1017–1029 (2012)
57. Qiu, W., Zheng, Z., Wang, X., Yang, X., Lyu, M.R.: Reputation-aware QoS Value prediction of web services. In: Proceedings of the 2013 IEEE 10th International Conference on Services Computing
58. Zheng, Z., Wu, X., Zhang, Y., Lyu, M.R., Wang, J.: QoS ranking prediction for cloud services. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1213–1222 (2013)
59. Zheng, Z., Ma, H., Lyu, M.R., King, I.: QoS-aware web service recommendation by collaborative filtering. Proc. IEEE Trans. Serv. Comput. **4**(2), 140–152 (2011)
60. Zheng, Z., Zhang, Y., Lyu, M.R.: Distributed QoS evaluation for real-world web services. In: Proceedings of the 2010 IEEE International Conference on Web Services
61. Arvelin, K.J., Kekalainen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. **20**(4), 422–446 (2002)

K. Jayapriya received her Bachelor of Computer Science degree from Sri Parasakthi College for Women College of Manonmaniam Sundaranar University in 1996 and the Master of Computer Application from Madurai Kamaraj University in 2002. She has completed her Ph.D. in Mother Teresa Women's University in 2013. She has over 18 years of experience in the IT field, Involved in the Development and Training of software applications in the Client/Server environment and Research environment. Her research interests include Image Processing, Pattern Recognition, Biometrics, Networking, Data Mining and Mobile Computing.

N. Ani Brown Mary received her B.E. degree in Computer Science and Engineering from The Infant Jesus College of Engineering, Tuticorin in 2009, and the M.E. degree in Computer and Information Technology from Anna University, Tirunelveli in 2013. She is working towards Ph.D. degree at the Information and Communication Engineering at Anna University, Chennai. Her research interests includes Image Processing, Data Mining and Cloud Computing.

R. S. Rajesh is presently working as a Professor in the Department of Computer Science and Engineering, M.S. University, Tirunelveli. He completed his B.E. in Electronics and Communication Engineering from Madurai Kamaraj University in the year 1988 and M.E. in Electronics and Communication Engineering from Madurai Kamaraj University in 1989. He has completed his Ph.D. in Computer Science and Engineering from M.S. University, Tirunelveli in 2004. He has 25 years of teaching experience and 20 years of research experience. He published 85 International Journals and 50 International conferences. He is also a full time recognized guide for various Universities. His areas of interest are Mobile Computing, Computer Networks, Image Processing, and Parallel Processing.